Combiner Optimisation Stochastique et Frontières pour l'Exploration 3D avec un Flotte de Drones

Alessandro Renzaglia, Jilles Dibangoye, Vincent Le Doze, Olivier Simonin

Combiner Optimisation Stochastique et Frontières pour l'Exploration 3D avec un Flotte de Drones

A. Renzaglia[a,b] alessandro.renzaglia@inria.fr
J. Dibangoye[b] jilles-steeve.dibangoye@insa-lyon.fr
V. Le Doze[b] vincent.le-doze@inria.fr
O. Simonin[b] olivier.simonin@insa-lyon.fr

[a]Univ. Grenoble Alpes, Inria, Chroma, F-38000 Grenoble, France.

[b]INSA Lyon, CITI Lab, Inria, Chroma, Lyon, France.

Résumé

Ce papier adresse le problème de l'exploration de terrains inconnus par une flotte de drones aériens coopératifs. Nous présentons une nouvelle approche décentralisée qui alterne exploration locale par optimisation stochastique et exploration par frontières. L'approche permet à chaque robot de générer une trajectoire en fonction des données qu'il collecte et de la carte locale qu'il construit par intégration des données partagées entre agents. Dès que l'agent arrive dans un minimum local, correspondant à une position où il est entouré d'espaces déjà explorés, alors l'algorithme identifie la plus proche frontière où il se rend avant de reprendre l'optimisation locale. Avec un faible coût calculatoire, une capacité à gérer les contraintes, et une prise de décision décentralisée, l'approche est particulièrement adaptée aux applications multi-robot en environnement complexes 3D. Les résultats en simulation montrent que l'approche génère des trajectoires sures et valides qui guident les robots pour une exploration complète de l'environnement. Par ailleurs, en terme de temps d'exploration, notre approche est significativement meilleure que la méthode des frontières proches. Elle fournit des temps équivalents à la méthode gloutonne centralisée tout en étant bien moins côuteuse en calcul.

Mots-clés : *Exploration multi-robot, cartographie 3D, optimisation stochastique locale, approche frontière.*

Abstract

This paper addresses the problem of exploring unknown terrains with a fleet of cooperating aerial vehicles. We present a novel decentralized approach which alternates gradient-free stochastic optimization and frontier-based approaches. Our method allows each robot to generate its trajectory based on the collected data and the local map built integrating the information shared by its teammates. Whenever a local optimum is reached, which corresponds to a location surrounded by already explored areas, the algorithm identifies the closest frontier to get over it and restarts the local optimization. Its low computational cost, the capability to deal with constraints and the decentralized decision-making make it particularly suitable for multi-robot applications in complex 3D environments. Simulation results show that our approach generates feasible and safe trajectories which drive multiple robots to completely explore realistic environments. Furthermore, in terms of exploration time, our algorithm significantly outperforms a standard solution based on closest frontier points while providing similar performances compared to a computationally more expensive centralized greedy solution.

Keywords: *Multi-robot exploration, 3D mapping, local stochastic optimization, frontier-based approach.*

1 Introduction

L'exploration d'environnements inconnus est une des tâches où l'usage de robots mobiles peut apporter une aide cruciale aux hommes dans des scénarios complexes et critiques. En particulier, le développement rapide des drones autonomes (UAV [1] en anglais) permet d'imaginer des missions de plus en plus complexes, comme l'exploration de vastes environnements inaccessibles pour les véhicules terrestres standards. Cependant, les solutions pour de tels scénarios se heurtent aux fortes contraintes des plateformes de petits drones : la limitation des capacités de calcul embarquées et des batteries. Cela nécessite d'explorer la définition de solutions très légères en calcul.

L'objectif de l'exploration autonome est de visiter (ou cartographier) complètement un envi-

1. Unmanned Aerial Vehicle

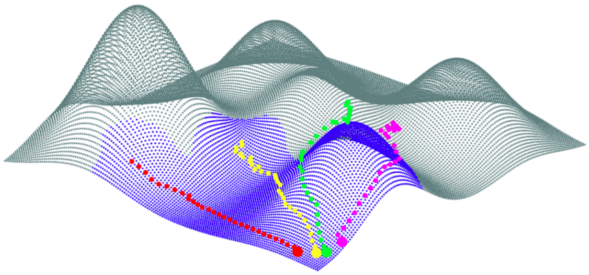

FIGURE 1 – Quatre drones débutant leur mission d'exploration d'un environnement inconnu

ronnement initialement inconnu en un minimum de temps (cf. Fig. 1). Ainsi, chaque robot doit décider de sa prochaine position afin de maximiser le gain sur la zone explorée. Comme dans toute mission multi-robot, la coordination entre agents joue un rôle crucial [19]. La problématique est de réduire les informations redondantes entre agents pour accélérer le processus d'exploration. L'approche standard pour le problème de l'exploration est fondée sur le concept de frontières, c'est-à-dire les cellules ou lignes séparant une région explorée d'un région encore inexplorée [18]. Toutefois, les solutions par frontières ne satisfont pas la contrainte pré-citée des petites plate-formes pour l'exploration aérienne d'environnements 3D. L'assignation des frontières aux robots requière une optimisation centralisée complexe et, même en considérant des stratégies heuristiques, elle demande à répéter régulièrement l'identification des frontières et le calcul des distances robot-frontières. Ce sont des opérations très lourdes en particulier pour des environnements 3D.

Pour dépasser ces limites, nous proposons une nouvelle approche où les robots sont guidés par une optimisation locale exploitant les données collectées pour maximiser l'espace nouvellement exploré, et une stratégie basée sur les frontières pour garantir la complétude de l'exploration. L'optimisation repose sur un algorithme d'optimisation stochastique exécuté indépendamment sur chaque robot et exploitant les informations de la carte construite par la flotte. L'optimisation étant strictement locale, les robots peuvent toutefois rester piégés dans des minima locaux avant de terminer leur tâche. Quand un agent détecte une telle situation, il exploite la carte globale pour identifier une frontière qu'il va rejoindre avant de reprendre son exploration locale. Il s'agit donc d'une approche hybride, combinant optimisation locale et frontières, qui vise à dépasser les limites de chaque technique et à construire une solution plus efficace fondée sur leurs avantages respectifs.

Le reste du papier est organisé comme suit. La prochaine section présente brièvement les travaux existants, avec une attention spéciale pour l'exploration multi-robot d'environnements non planaires. En Section 3, nous formalisons le problème de l'exploration multi-robot comme un problème d'optimisation. La Section 4 présente en détail l'approche proposée, en décrivant l'optimisation locale adoptée et la méthode fondée sur les frontières. Enfin, en Section 5 des expérimentations en simulation sont présentées pour évaluer l'algorithme proposé.

2 Travaux existants

Le problème de l'exploration autonome a été considérablement étudié ces dernières années, rendant difficile la synthèse d'une bibliographie riche. Une tentative de synthèse proposée par [9], offre une analyse comparative étendue des principales méthodes existantes. Une autre revue de la litérature proposée par [1] s'est focalisée sur les aspects liés à la communication.

La pluspart des méthodes existantes pour l'exploration autonome sont basées sur le concept de frontières. Ce concept a été introduit initialement par Yamauchi d'abord dans le cadre mono-robot [17] puis dans le cadre multi-robot [18]. Cette idée simple mais efficace a donné lieu à de nombreuses variantes et améliorations dans les années suivantes, en particulier dans le cadre des environnements 2D. Une approche centralisée, efficace mais coûteuse, a été introduite par [5] afin de résoudre un problème multi-critère considérant à la fois les coûts et les gains associés aux trajets. L'algorithme MINPOS [3] propose une approche d'allocation décentralisée des points frontières suivant un ordre sur les membres de la flotte de robots. Cet ordre, basé sur la distance à chaque frontière, permet d'obtenir une distribution spatiale homogène des robots dans l'environnement. Dans [7], les auteurs proposent une reformulation du problème d'exploration par frontières comme plusieurs problèmes de voyageur de commerce, c'est à dire un pour chaque robot et les points frontières encore disponibles. Bien que cette stratégie d'exploration conduise à un temps d'exploration plus court par rapport à d'autres stratégies, son coût calculatoire exorbitant rend son application limitée, contrairement à notre proposition. Une approche alternative, pour faire face au problème d'allocation des

points frontières de façon distribuée, repose sur une reformulation en un problème de satisfaction de contraintes distribuée (DSCP), comme proposée dans [13]. Au-délà du problème d'allocation des points frontières, un problème tout aussi important est celui portant sur le besoin de communication durant la tâche. Cette problématique est notamment addressée dans [2]. Les auteurs y présentent une stratégie visant à coordonner les robots entre eux afin de former un réseau capable de satisfaire les contraintes de connectivité.

Dans le cadre des environnements 3D, une extension directe de l'approche basée frontières a été proposée dans [6]. Comme mentionné précédemment, en trois dimensions, le processus d'identification des points frontières est bien plus complexe, et a donné lieu à quelques travaux, eg. [20]. Dans un cadre simulé, [12] présente une approche par frontières centralisée visant à explorer des scènes tridimensionnelles avec une flotte de micro-véhicles aériens munis de caméras. Notre approche se distingue par une solution distribuée, et par des communications entre robots qui se limitent à l'échange des points frontières et non des cartes.

La solution que nous proposons dans cet article a pour objectif de dépasser le recours systématique au calcul des points frontières à chaque pas de décision. Il s'agit de réduire le coût total de calcul. Le recours aux points frontière est significativement réduit par la combinaison avec un algorithme d'optimisation locale décentralisé. La solution résultante guide les robots vers une position maximisant l'espérance du gain d'information, tout en assurant que la tâche d'exploration sera complètement réalisée.

3 Exploration 3D multi-robot

Le problème de planification multi-robot considéré dans ce papier porte sur l'exploration autonome, par une flotte de drones aériens, d'un terrain inconnu représenté par une surface $\mathcal{S}$ dans un environnement 3D, en un temps minimum. Formellement, ce problème est défini par un tuple $(N, \mathcal{S}, \mathcal{P}, \mathcal{C}, E)$ où :

- N est le nombre de robots $r \in \{1, 2, \ldots, N\}$;
- $\mathcal{S}$ est un ensemble *inconnu* de points 3D (voxels) notés $s \in \mathcal{S}$;
- $\mathcal{P}$ est l'ensemble des positions $\mathbf{P}$, où $\mathbf{P}_t^{(r)} \doteq [x_{1,t}^{(r)}, x_{2,t}^{(r)}, x_{3,t}^{(r)}]$ est la position du drone r à l'étape t ;
- $\mathcal{C}(\mathbf{P}^{(r)}) \preceq 0$ défini les contraintes *partiellement inconnues* que les positions $\mathbf{P}^{(r)}$ de chaque drone doit satisfaire afin d'éviter les collisions drone-obstacle et drone-drone ainsi que la satisfaction des contraintes portant sur l'altitude minimale et maximale de vol ;
- $E_t(S)$ est la portion de la surface S explorée jusqu'à l'instant t. Un point s est considéré exploré par un agent si : (1) ils sont connectés par la ligne de vue et (2) ils sont à une distance plus petite que la portée maximale du champs de vision.

Soit $e_t(s)$ le sous-ensemble d'agents qui explore pour la première fois le voxel s à l'instant t, c-à-d

$$e_t(s) = \{r \in \mathcal{R} \mid s \in E_t(S) \backslash E_{t-1}(S), O(s, \mathbf{P}_t^{(r)}) = 1\}$$

où $O(s, \mathbf{P})$ est vrai lorsque le point s est observé par un agent positionné en $\mathbf{P}$. De plus, les points frontières sont définis comme les voxels déjà explorés $s \in E(\mathcal{S})$ qui sont adjacents aux régions non-explorées, c-à-d $\mathcal{S} \backslash E(\mathcal{S})$.

L'objectif de ce problème est de trouver les trajectoires des robots $(\mathbf{P}_t^{(r)})_{r=1,\ldots,N;t=1,\ldots,t}$ qui permettent d'explorer la totalité de la surface S en un temps minimum T, tout en satisfaisant les contraintes $\mathcal{C}$. Ce problème est impossible à résoudre hors-ligne étant donné que l'environnement est initialement inconnu ; ainsi une solution en ligne, même sous-optimale, est nécessaire. Pour y parvenir, nous définissons pour chaque robot r une fonction d'optimisation comme suit :

$$J_t^{(r)} = \sum_{s \in E_t(\mathcal{S})} \frac{\delta_s^r}{|e(s)|}, \quad \delta_s^r = \begin{cases} 1 & \text{if} \quad r \in e(s) \\ 0 & \text{if} \quad r \notin e(s) \end{cases} \tag{1}$$

Afin de maximiser cette fonction, chaque drone tente à chaque itération de maximiser la quantité de nouveaux voxels s (composante δ_s^r) explorés, tout en minimisant les informations redondantes (c-à-d l'exploration simultanée de nouvelles régions par plus d'un drone). Ceci est dû au terme $|e(s)|$, qui représente la pénalisation utile pour réduire l'importance des voxels qui ont été explorés simultanément par plusieurs drones. En d'autres mots, chaque voxel a une récompense fixée qui est répartie sur l'ensemble des agents qui l'ont vu en premier. Cette récompense a pour valeur maximale 1, lorsqu'un seul agent l'a exploré. Cette stratégie d'attribution des récompenses vise à forcer la répartition des agents, afin de minimiser la redondance des informations collectées. À noter que la somme sur l'ensemble des drones des fonctions $J_t^{(r)}$ est simple-

ment le nombre de voxels exploré jusqu'à l'instant t, c-à-d

$$\sum_{r\in N} J_t^{(r)} = \sum_{r\in N}\sum_{s\in E_t(\mathcal{S})} \frac{\delta_s^r}{|e(s)|} = |E_t(\mathcal{S})|\,. \quad (2)$$

4 L'algorithme FCAO

Dans cette section, nous présentons un nouvel algorithme d'exploration multi-robots combinant une optimisation stochastique basée sur une approximation locale de la fonction objectif avec une approche basée sur les frontières. Nous nommons cet algorithme Frontier-based Cognitive Adaptive Optimization (FCAO) (Algorithme 1). L'approche décentralisée qui en résulte a pour objectif de surmonter les faiblesses des deux méthodes utilisées tout en préservant leurs points forts.

Algorithm 1: FCAO

```
function FCAO(w, α, E)
    Initialiser t, δt = [t − w : t] and P_δt.
    repeat
        CAO(t, P_δt, E, α)
        P_{t+1} ← ClosestFrontier(P_t, E)
        MoveTo(P_{t+1})
        t ← t + 1
    until P_t = P_{t−1};
function CAO(t, P_δt, E, α)
    repeat
        Maj J(E) avec nouveaux points
        Maj ϑ_t selon équation (5).
        Générer points (ζ^(m))_{m∈{1,...,M}}
          aléatoirement.
        Sélectionner m* selon équation (7)
        P_{t+1} ← P_t + αζ^(m*)
        MoveTo(P_{t+1})
        t ← t + 1
    until P_t = P_{t−1};
function ClosestFrontier(P_t, E)
    F = CandidateFrontiers(E)
    return SelectClosestFrontier(F, P_t)
```

4.1 Optimisation stochastique basée sur une approximation locale

Trouver un vrai optimum de (1) est impossible en pratique, car l'environnement est initialement inconnu et l'optimisation doit être poursuivie en ligne pendant le processus d'exploration. Cependant, chaque robot peut calculer les valeurs numériques de (1) à chaque pas de temps en fonction des données collectées. Dans ce cadre, les algorithmes d'optimisation stochastique basés sur une approximation locale de la fonction objectif d'origine sont des outils puissants pour contourner ce problème. C'est le cas, par exemple, de l'algorithme CAO, développé et analysé à l'origine par Kosmatopoulos dans [11], qui a récemment été proposé, dans diverses versions, pour trouver des solutions à des problèmes multi-robots, tels que : le déploiement optimal de plusieurs robots [14], [15], la détection de cible [16] et la localisation et cartographie multi-robots [10]. Dans cette section, nous présentons une version modifiée de cet algorithme d'optimisation pour le rendre approprié au problème considéré.

La solution proposée représente un moyen efficace de résoudre en temps réel des problèmes d'optimisation sous contrainte où la forme exacte de la fonction objectif $\mathcal{J}$ n'est pas disponible, mais dont les valeurs numériques peuvent être récupérées en exploitant les informations collectées. L'algorithme CAO est basé sur deux étapes principales : *i)* à chaque itération, une fonction d'approximation est construite pour avoir une estimation locale de la fonction objectif inconnue J ; *ii)* la position suivante du robot est sélectionnée de manière à maximiser la fonction d'approximation en respectant les contraintes du problème. Ces deux étapes sont détaillées ci-dessous.

1. A chaque instant t, la fonction J est estimé comme suit :

$$\hat{J}_t^{(r)}\left(x_{1,t}^{(r)}, x_{2,t}^{(r)}, x_{3,t}^{(r)}\right) = \vartheta_t^T\,\phi\left(x_{1,t}^{(r)}, x_{2,t}^{(r)}, x_{3,t}^{(r)}\right) \quad (3)$$

où ϑ_t désigne le vecteur des paramètres estimés, calculés aux instants t, et ϕ le vecteur des fonctions de régression. Dans notre cas, nous avons choisi cette fonction ϕ comme un polynôme de troisième degré des variables d'état, c'est-à-dire :

$$\phi = 1 + \sum_{i=1}^{3} x_{i,t}^{(r)} + \sum_{i=1}^{3}\sum_{j\geq i}^{3} x_{i,t}^{(r)} x_{j,t}^{(r)} + \sum_{i=1}^{3}\sum_{j\geq i}^{3}\sum_{k\geq j}^{3} x_{i,t}^{(r)} x_{j,t}^{(r)} x_{k,t}^{(r)}\,. \quad (4)$$

Ce choix est arbitraire et d'autres solutions peuvent également être adoptées. Cependant, il convient de noter que cette approximation est strictement locale et qu'elle n'a aucune ambition

de reproduire la fonction originale dans tout l'espace d'exploration. Son rôle est exclusivement de fournir une estimation de J pour permettre à chaque robot de sélectionner sa prochaine position. Le vecteur des paramètres d'estimation ϑ_t, de dimension 20 dans notre cas, est ensuite calculé à chaque instant t sur la base d'un ensemble limité de mesures antérieures comme suit :

$$\vartheta_t = \underset{\vartheta}{\operatorname{argmin}} \frac{1}{2} \sum_{\ell=\ell_t}^{t-1} \left(J_\ell^{(r)} - \vartheta^T \phi \left(\mathbf{P}_\ell^{(r)} \right) \right)^2 \quad (5)$$

où $\ell_t = \max\{0, t - L - T_h\}$, avec T_h étant un entier non négatif défini par l'utilisateur, et $J_\ell^{(r)}$ la valeur numérique de la fonction objectif au temps ℓ. Des algorithmes standard d'optimisation des moindres carrés peuvent être utilisés pour résoudre (5). Notez que l'utilisation d'un ensemble limité de valeurs antérieures présente l'avantage de réduire considérablement le temps de calcul nécessaire pour obtenir le vecteur ϑ_t, mais la fonction résultante est une approximation fiable uniquement localement et ne peut pas être utilisée pour une optimisation globale.

2. Dès que l'estimateur $\hat{J}_t$ est construit selon (3) et (5), la nouvelle position du robot est obtenue par une recherche aléatoire. Un ensemble de nouvelles positions admissibles est généré en perturbant de manière aléatoire l'état actuel et directement testé sur $\hat{J}_t$. Plus formellement, un ensemble de M configurations d'états candidats est construit selon :

$$\mathbf{P}_t^{(r,m)} = \mathbf{P}_t^{(r)} + \alpha\, \zeta_t^{(r,m)}, \forall m \in \{1, \ldots, M\}, \quad (6)$$

où $\zeta_t^{(r,m)}$ est un vecteur aléatoire unitaire de moyenne zéro avec une dimension égale à celle de $\mathbf{P}_t^{(r)}$ et α représente le pas d'exploration pour chaque itération. Notez que ce paramètre, dans les algorithmes d'optimisation, dépend généralement du temps pour assurer la convergence vers un optimum local (voir, par exemple, [4]). Dans notre cas, nous n'avons pas cette nécessité puisque dans une tâche d'exploration, il n'y a aucun intérêt à se déplacer avec un pas inférieur au pas maximal réalisable et il n'y a pas de convergence tant que la totalité de l'environnement n'est pas explorée. Parmi toutes les M nouvelles configurations candidates $\mathbf{P}_t^{r,m}$, celles qui correspondent à des positions non réalisables, c'est-à-dire celles qui violent les contraintes $\mathcal{C}$, sont négligées et la nouvelle position du robot sera $\mathbf{P}_{t+1}^{(r)} = \mathbf{P}_t^{(r,m^*)}$ où m^* est calculé comme suit :

$$m^* = \underset{m \in \{1,\ldots,M\}}{\operatorname{argmax}} \hat{J}_t^{(r)} \left(\mathbf{P}_t^{(r,m)} \right). \quad (7)$$

Le choix aléatoire des candidats est essentiel et crucial pour l'efficacité de l'algorithme, car ce choix garantit que $\hat{J}_t$ soit une estimation fiable et précise de la fonction inconnue J. Enfin, précisons que, dans cet article, nous nous concentrons uniquement sur le problème de planification de haut niveau sans prendre en compte explicitement les contraintes de dynamique pour les robots. Ce choix peut également être justifié en supposant que les nouveaux points de déplacement sont suffisamment éloignés pour être toujours accessibles par le robot. Cependant, des hypothèses plus restrictives, y compris sur la dynamique du robot, réduisant l'espace des prochaines positions possibles n'affecteraient que la définition de $\mathcal{C}$, mais pas l'algorithme.

4.2 Plus proche frontière (closest frontier)

Comme indiqué précédemment, l'optimisation proposée est strictement locale et, même si elle peut trouver très efficacement une solution admissible à un problème de génération de trajectoire sous contrainte, elle nécessite de mesurer des variations dans les valeurs de la fonction d'optimisation pour fournir de meilleures solutions à chaque itération. En d'autres termes, comme chaque méthode d'optimisation locale, elle ne peut pas surmonter les optima locaux sans informations externes. Dans une tâche d'exploration, cela signifie que si les mouvements du robot apportent de nouvelles zones explorées, l'algorithme fournit une direction claire à suivre, mais lorsqu'il reste entouré de régions déjà explorées, la méthode d'optimisation ne peut fournir qu'un mouvement aléatoire. Ce n'est que dans ce cas-là que nous calculons l'ensemble des frontières disponibles et que le robot doit se déplacer à la plus proche. Cela nécessite de calculer chaque distance drone-frontière dans l'espace 3D, puis de sélectionner la plus petite (adaptation de l'approche standard [17] aux espaces 3D). Une telle attribution de frontière permet au robot de surmonter efficacement les blocages, acquérir de nouvelles informations et relancer l'optimisation locale à partir de celle-ci. Plus globalement, alterner explorations locales et assignations de frontières permet de garantir l'exhaustivité de l'exploration, comme pour les autres méthodes fondées sur les frontières.

Cette étape utilise une carte globale que chaque

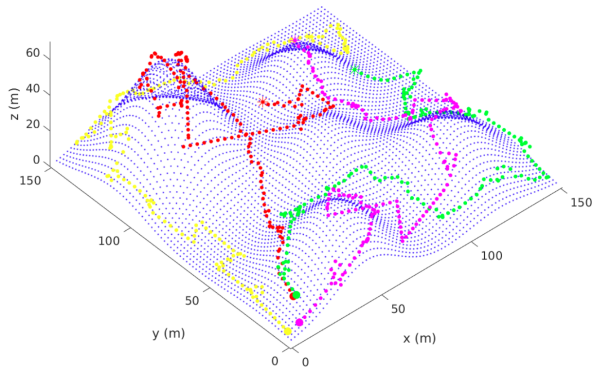

FIGURE 2 – Trajectoires réalisées par quatre robots lors de l'exploration d'un terrain inconnu.

robot construit en intégrant les informations reçues de ses coéquipiers. Nous nous référons à [18] pour une description plus détaillée de cette stratégie et pour les détails sur la communication nécessaire à la construction de cette carte.

L'aspect important de cette approche est que, même si l'information est partagée, la décision reste décentralisée. Cela garantit en particulier la robustesse vis-à-vis de l'échec des robots car, si un robot cessait de communiquer, les robots restants continueraient à explorer tout l'environnement. De plus, en tant que système complètement asynchrone, les robots n'ont pas besoin d'attendre d'autres robots et la perte d'un ou de plusieurs robots ne provoque aucune interruption de l'algorithme.

Le désavantage de cette approche est la possibilité d'une redondance dans l'exploration, c'est-à-dire que plusieurs robots peuvent être affectés à la même frontière. Cet inconvénient existe dans notre approche mais il est largement limité par le fait que les frontières ne sont utilisées que rarement et de manière asynchrone par les agents.

5 Résultats de simulations

L'approche proposée a été testée à partir de simulations puis comparée aux solutions standards de la littérature. Les terrains à explorer sont générés selon une répartition aléatoire de courbes gaussiennes variants en hauteur et en écart type (cf. exemple Fig. 2). Les robots doivent respecter une hauteur minimale relative au sol au cours du vol ainsi qu'une distance de sécurité avec les autres robots afin d'éviter toute collision. Nous avons fait comme hypothèse de simulation que chaque robot dispose d'une capacité de détection omnidirectionnelle avec une portée limitée.

La figure 2 illustre un exemple de mission d'exploration réalisée par 4 robots en montrant les trajectoires réalisées. Pour tous les résultats de simulation répertoriés dans cette partie les positions initiales des robots peuvent légèrement varier, elles sont cependant toujours proches les unes des autres dans un des coins de la zone à explorer.

Afin d'évaluer quantitativement les performances de l'approche proposée nous la comparons à deux autres techniques standards : *i)* un algorithme décentralisé d'exploration de la plus-proche frontière, dans lequel chaque robot sélectionne indépendemment de ses coéquipiers le point frontière le plus proche de lui-même (noté closest frontier) ; *ii)* un algorithme centralisé du type glouton, qui assigne - séquentiellement - a chaque robot un point frontière unique prélevé d'une liste (noté greedy). Dans ce dernier, les points frontières sont tout d'abord séparés en différents groupes pour réduire au maximum le chevauchement des trajectoires des robots, puis seuls les centroides de ces groupes sont pris en comptes.

Un premier résultat avec 8 robots, démarrant du même ensemble de positions initiales, est présenté figure 3. Dans un premier temps, nous pouvons observer que notre approche est bien plus performante que l'algorithme closest-frontier pour réaliser l'exploration complète. Ceci se justifie par le haut niveau de redondance de l'exploration guidée uniquement par la sélection des points frontières les plus proches. Dans notre approche, l'optimisation locale déploie efficacement l'équipe en évitant le recoupements des informations acquises. Dans un deuxième temps, nous pouvons voir que malgré son aspect décentralisé notre méthode est capable de réaliser une performance très proche de la méthode gloutonne qui est centralisée.

Dans le même scénario, nous étudions un autre aspect important : le temps de calcul nécessaire pour trouver la nouvelle position objectif d'un robot à chaque itération de l'algorithme. Comme indiqué précédemment, l'un des point fort de notre approche est le faible coût calculatoire. Celui-ci est clairement illustré par la figure 3 (en bas), avec un facteur temps non seulement significativement inférieur au temps requis par l'algorithme glouton mais aussi meilleur que celui de l'algorithme très simple de plus proche frontière. La différence principale se situe dans le besoin de l'approche fondée frontière de calculer à chaque itération les distances entre la position du robot et celles de tous les points

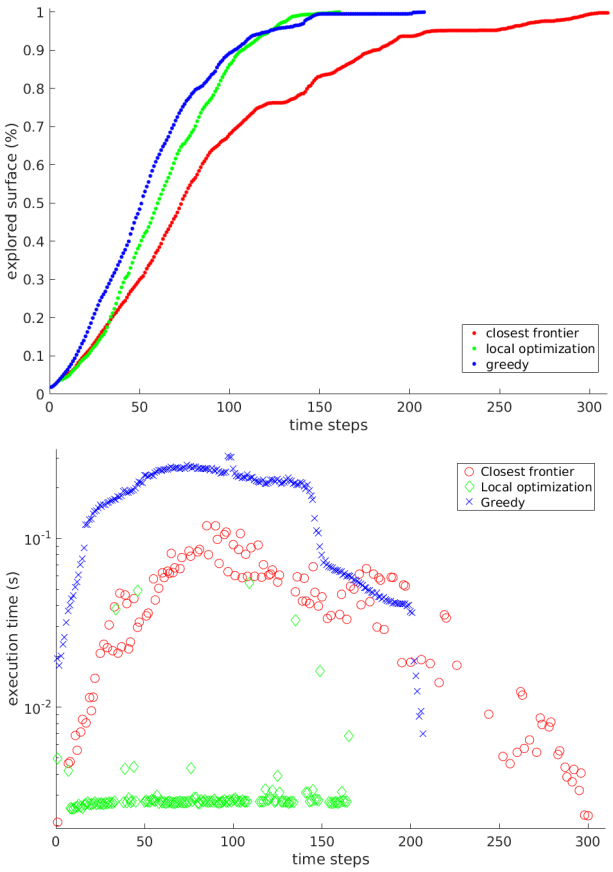

FIGURE 3 – L'approche proposée est comparée à deux alternatives : L'algorithme décentralisé de sélection de la plus proche frontière (closest) et l'algorithme centralisé d'affectation gloutonne des points frontières (greedy). Les résultats correspondent à un scénario avec 8 robots évoluant dans un environnement similaire à celui de la figure 2. En haut : Portion de l'environnement exploré en fonction du nombre de pas de temps. En bas : Temps de calcul mis pour sélectionner la position suivante à chaque itération.

frontières, ce qui peut devenir une opération très coûteuse en 3D. Dans notre cas l'effet est significatif, même si nous ne calculons pas la distance géodésique exacte entre deux points, ce qui rendrait les écarts encore plus grands. A la place nous calculons la ligne directe qui relie deux points en survolant le terrain avec une altitude respectant une contrainte de hauteur minimale par rapport au sol. Le coût de notre approche est vraiment indépendant du nombre de points frontières, comme le montre la courbe 3, où le temps d'exécution de l'optimisation locale est constant (sauf pour les rares cas ou les frontières sont exploitées) tandis que deux autres courbes varient au cours du temps à cause de leur dépendance au nombre de frontières présentes sur la

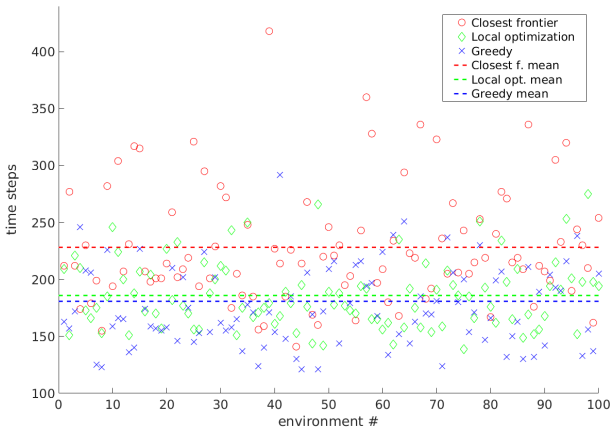

FIGURE 4 – Temps d'exploration des trois algorithmes obtenus sur 100 environnements générés aléatoirement. Les lignes en pointillés représentent la performance moyenne de chaque approche.

carte à chaque instant. De plus, en étant décentralisé, son coût est aussi indépendant du nombre de robots, contrairement à l'algorithme centralisé glouton. Le seul coût significatif de l'optimisation locale provient de la résolution de (5), qui dépend exclusivement de la dimension de la fonction ϕ et du nombre de mesures considérées dans l'historique, qui sont tout deux constants au cours de la mission.

Afin d'avoir des résultats statistiquement plus significatifs, nous avons comparé les trois méthodes sur 100 environnements générés aléatoirement (8 sommets avec des écarts-type égaux et centrés sur des positions tirées aléatoirement avec une distribution uniforme sur l'environnement). Les résultats présentés en figure 4 donnent le temps d'exploration final pour chaque essais ainsi que la moyenne de chaque algorithme. Comme dans la précédente étude, l'approche proposée par optimisation locale montre une performance très proche de celle de l'algorithme centralisé glouton. Ces deux dernières sont par ailleurs bien plus performantes que la méthode plus proches frontière. En plus d'avoir un fonctionnement moins performant en moyenne, cette dernière méthode présente aussi un écart-type plus important sur les résultats, alors que les deux autres montrent une bien meilleure robustesse par rapport aux différences entre les environnements.

5.1 Environement urbain simulé

Nous présentons maintenant les résultats obtenus avec une modélisation réaliste de l'environ-

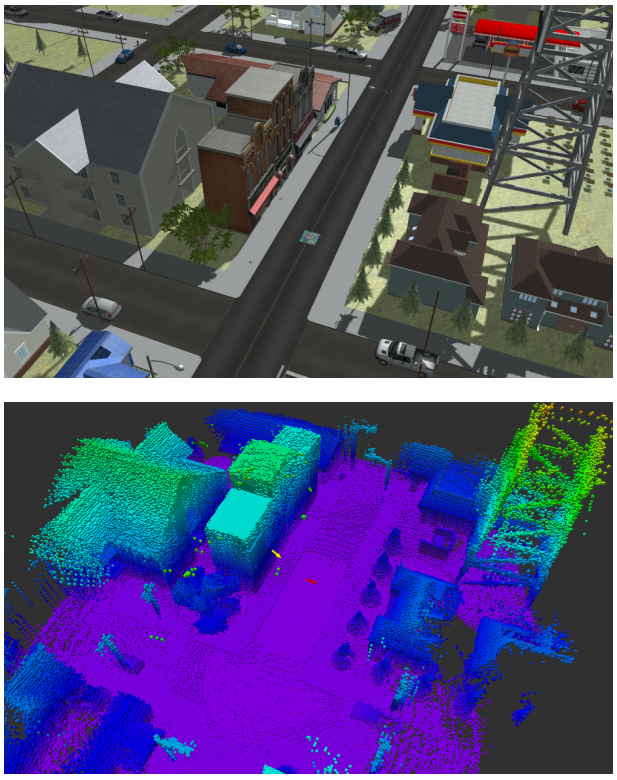

FIGURE 5 – Environnement à explorer (en haut) et carte 3D finale (en bas) construite par 2 drones exécutants l'algorithme FCAO.

nement et des drones afin d'illustrer la faisabilité et l'intéret de notre méthode pour des applications concrètes. Les simulations ont été réalisées avec les outils ROS et Gazebo, qui permettent de générer un contexte urbain riche et réaliste contenant des batiments, des infrastructures, des véhicules, etc. (cf. figure 5). La modélisation des drones a été réalisée en prenant comme référence les spécifications du quadrirotor Intel Aero. Chaque drone embarque un capteur lui permettant d'obtenir en temps réel une représentation spatiale sous forme de nuage de points de l'environement situé en dessous de lui dans une demi-sphère de rayon 20m. La cartographie de la surface explorée ainsi que la liste des coordonnées des frontières sont générés, lorsque cela est necessaire, à partir des nuages de points grâce au package ROS Octomap_server [8]. Les captures d'écran de la figure 5 illustrent l'exploration réalisée par deux drones simulés en montrant l'environnement tel que modélisé par Gazebo et la carte finale représentée par Octomap. La performance globale est illustrée par la courbe 6 où le nombre de noeuds de l'Octree est tracé en fonction du temps d'exploration des deux quadrirotors. Une vidéo explicative montrant le déroulement de la simulation est disponible au lien ci-dessous [2].

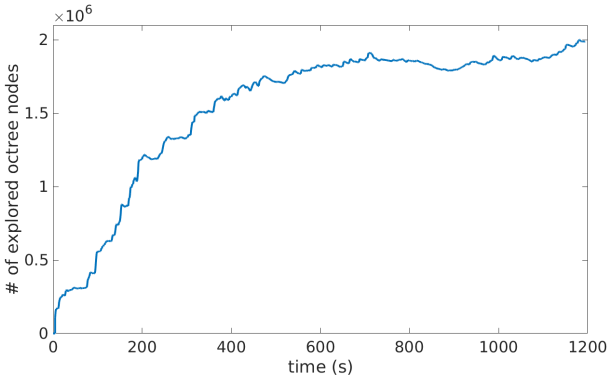

FIGURE 6 – Nombre de noeuds *octree* explorés au fil du temps dans le scénario décrit en Fig. 5.

6 Conclusions

Dans ce travail, nous avons présenté une nouvelle approche décentralisée pour l'exploration de terrains 3D avec une flotte de drones aériens, basée sur une optimisation locale de l'information acquise. L'optimisation est réalisée en adoptant un algorithme stochastique qui, avec un coût de calcul très bas, permet d'optimiser la fonction objectif uniquement sur la base des données collectées par les robots. Cette approche a été combinée avec une méthode fondée sur les frontières pour permettre de relancer l'optimisation lorsqu'un robot reste bloqué dans un optimum local. Des résultats en simulation ont prouvé l'efficacité de cette approche et une comparaison avec des approches standards basées sur les frontières a été réalisée. Cette analyse a montré que notre approche améliore considérablement les performances de algorithme décentralisé "plus proche frontière", tout en offrant des performances similaires en ce qui concerne une assignation de frontières centralisée et gloutonne beaucoup plus coûteuse. Une dernière expérimentation avec le simulateur Gazebo montre également l'applicabilité de notre approche dans un scénario plus réaliste.

Les travaux futurs porteront sur une validation plus approfondie de notre algorithme dans des environnements plus riches et variés, ainsi que sur un modèle plus réaliste des drones en prévision d'expériences dans des scénarios réels.

2. https://team.inria.fr/chroma/files/2019/05/3DMulti-UAVExploration.mp4

Références

[1] Francesco Amigoni, Jacopo Banfi, and Nicola Basilico. Multirobot exploration of communication-restricted environments : A survey. *IEEE Intelligent Systems*, 32(6) :48–57, 2017.

[2] Jacopo Banfi, Alberto Quattrini Li, Ioannis Rekleitis, Francesco Amigoni, and Nicola Basilico. Strategies for coordinated multirobot exploration with recurrent connectivity constraints. *Autonomous Robots*, 42(4) :875–894, 2018.

[3] Antoine Bautin, Olivier Simonin, and François Charpillet. Minpos : A novel frontier allocation algorithm for multi-robot exploration. In *International conference on intelligent robotics and applications*, pages 496–508. Springer, 2012.

[4] Dimitri Bertsekas and John Tsitsiklis. Gradient convergence in gradient methods with errors. *Journal on Optimization*, 10(3), 2000.

[5] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E Schneider. Coordinated multi-robot exploration. *IEEE Transactions on robotics*, 21(3) :376–386, 2005.

[6] Christian Dornhege and Alexander Kleiner. A frontier-void-based approach for autonomous exploration in 3d. *Advanced Robotics*, 27(6) :459–468, 2013.

[7] Jan Faigl, Miroslav Kulich, and Libor Přeučil. Goal assignment using distance cost in multi-robot exploration. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3741–3746. IEEE, 2012.

[8] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap : An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34(3) :189–206, 2013.

[9] Miguel Juliá, Arturo Gil, and Oscar Reinoso. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33(4) :427–444, 2012.

[10] Athanasios Kapoutsis, Savvas Chatzichristofis, Lefteris Doitsidis, Joao Borges de Sousa, Jose Pinto, Jose Braga, and Elias B Kosmatopoulos. Real-time adaptive multi-robot exploration with application to underwater map construction. *Autonomous robots*, 40(6) :987–1015, 2016.

[11] Elias B Kosmatopoulos. An adaptive optimization scheme with satisfactory transient performance. *Automatica*, 45(3), 2009.

[12] Nesrine Mahdoui, Vincent Frémont, and Enrico Natalizio. Cooperative frontier-based exploration strategy for multi-robot system. In *2018 13th Annual Conference on System of Systems Engineering (SoSE)*, pages 203–210. IEEE, 2018.

[13] Pierre Monier, Arnaud Doniec, Sylvain Piechowiak, and René Mandiau. Comparison of dcsp algorithms : a case study for multi-agent exploration. In *Advances on Practical Applications of Agents and Multiagent Systems*, pages 231–236. Springer, 2011.

[14] Alessandro Renzaglia, Lefteris Doitsidis, Agostino Martinelli, and Elias B Kosmatopoulos. Multi-robot three-dimensional coverage of unknown areas. *The International Journal of Robotics Research*, 31(6), 2012.

[15] Davide Scaramuzza et al. Vision-controlled micro flying robots : from system design to autonomous navigation and mapping in GPS-denied environments. *IEEE Robotics & Automation Magazine*, 21(3), 2014.

[16] Kuo-Shih Tseng and Bérénice Mettler. Near-optimal probabilistic search via submodularity and sparse regression. *Autonomous Robots*, 41(1), 2017.

[17] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 146–151. IEEE, 1997.

[18] Brian Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, pages 47–53. ACM, 1998.

[19] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10(12) :399, 2013.

[20] Cheng Zhu, Rong Ding, Mengxiang Lin, and Yuanyuan Wu. A 3d frontier-based exploration tool for mavs. In *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 348–352. IEEE, 2015.